在线分销产品
根据航班号查询航班可用舱位服务
用户手册

目录

1.概述

根据航班号查询航班可用舱位服务用来查询指定航班全舱位实时航班信息。该服务提供了简单有效的实时在线查询通道，可查询指定日期及航班号的可用航班信息。

2.版本

主要内容	根据航班号查询航班可用舱位
参考文档	
创建部门	

批准单		
版本号：		
批准人	批准人签字	批准日期

文档控制

版本	修改日期	修改内容描述	作者	批准人	批准日期
V1.0.0	2013-07-15	创建	谢杰		

3.业务范围

- 指定航班号的航班可用舱位查询

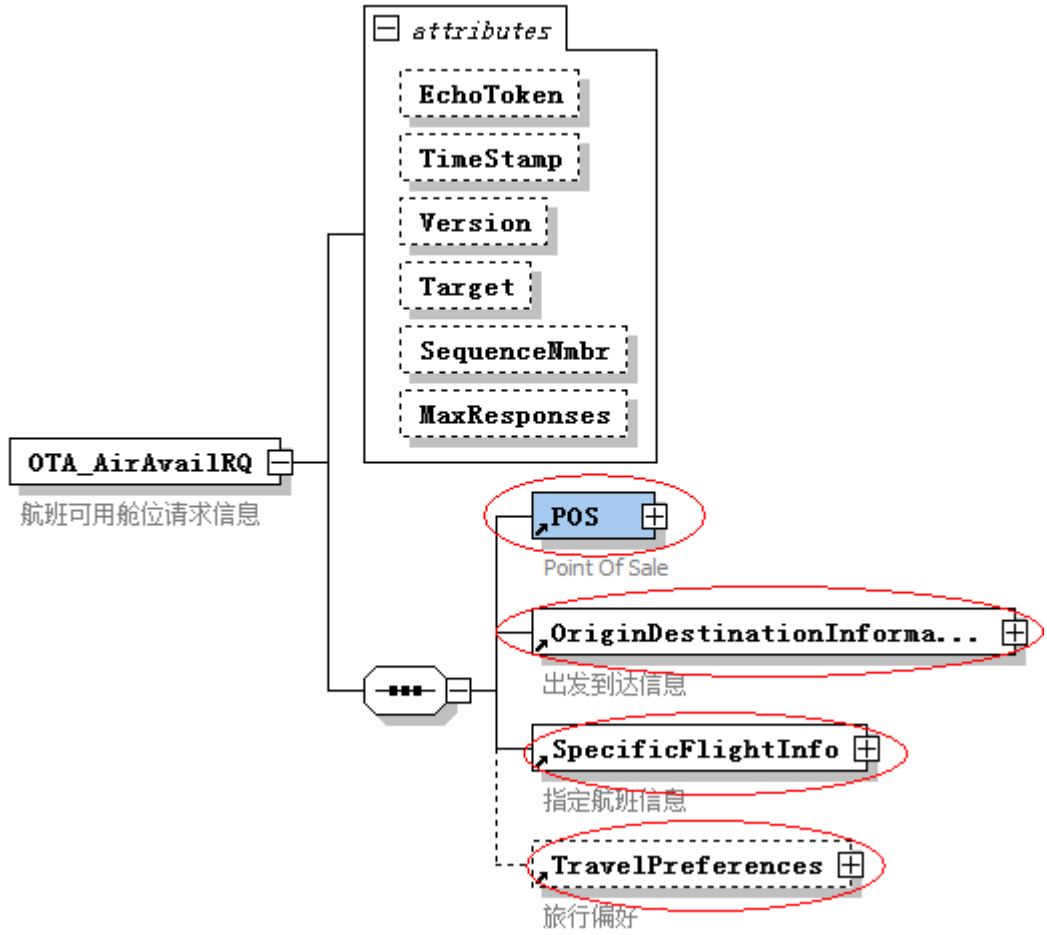
4.功能范围

用于查询航班座位可利用情况，及其相关航班信息如：航班号、舱位等。

5.接口说明

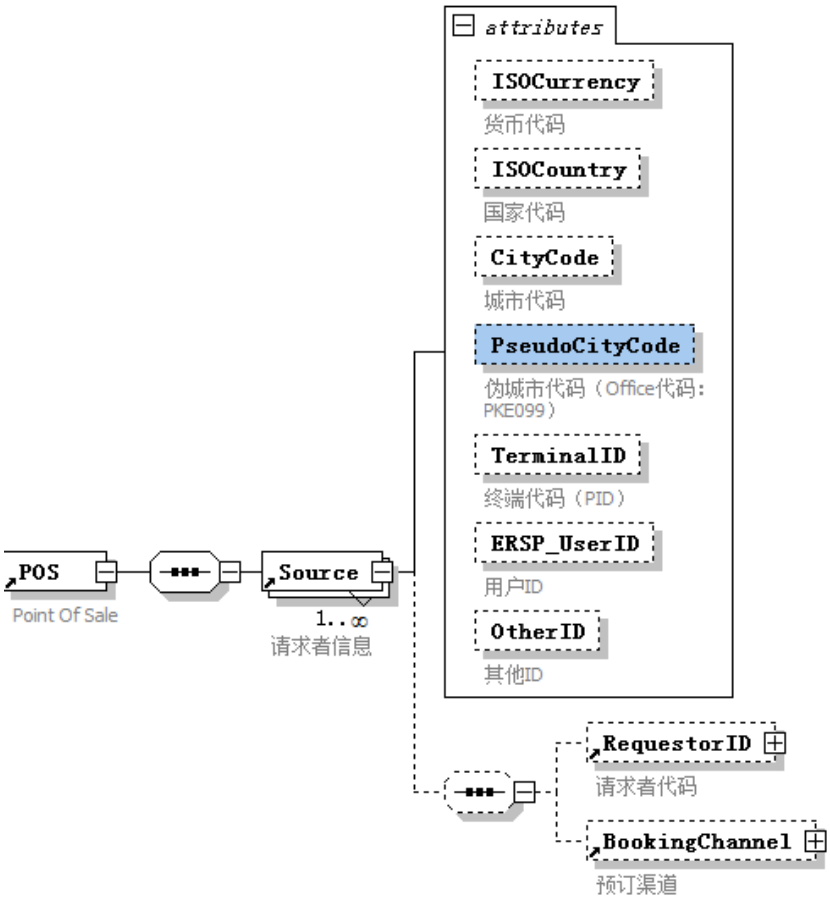
5.1 接口结构

5.1.1 请求结构

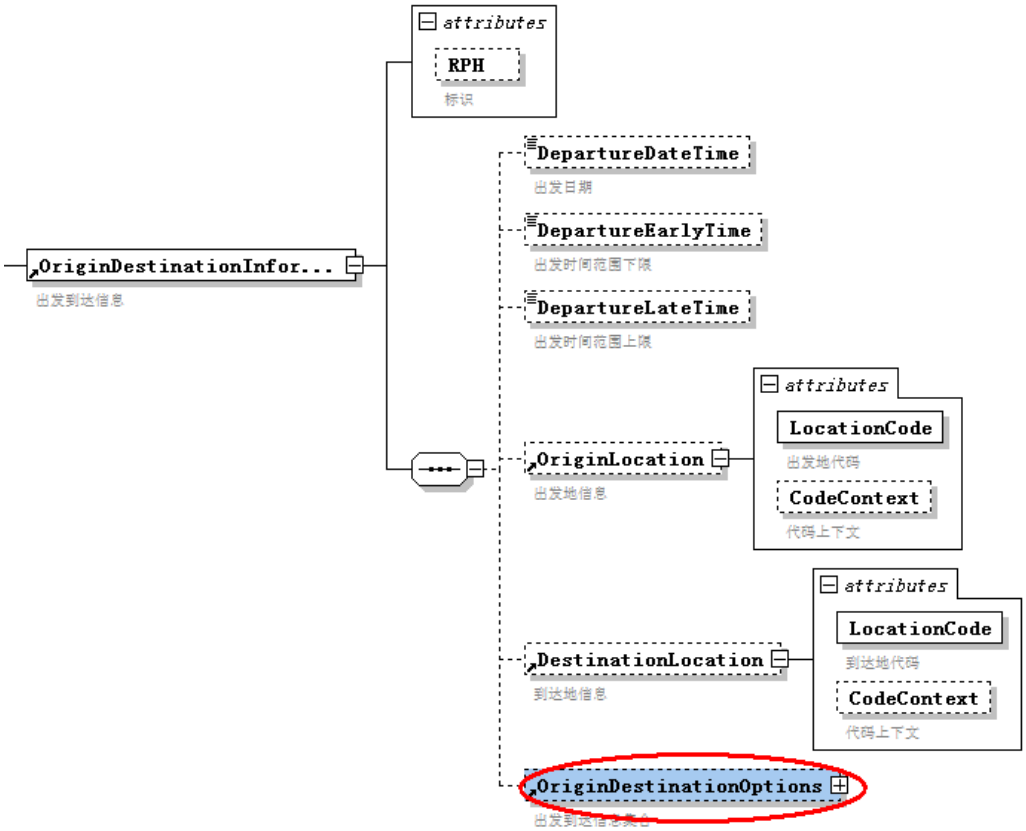


其中标识红色椭圆线的节点结构如下：

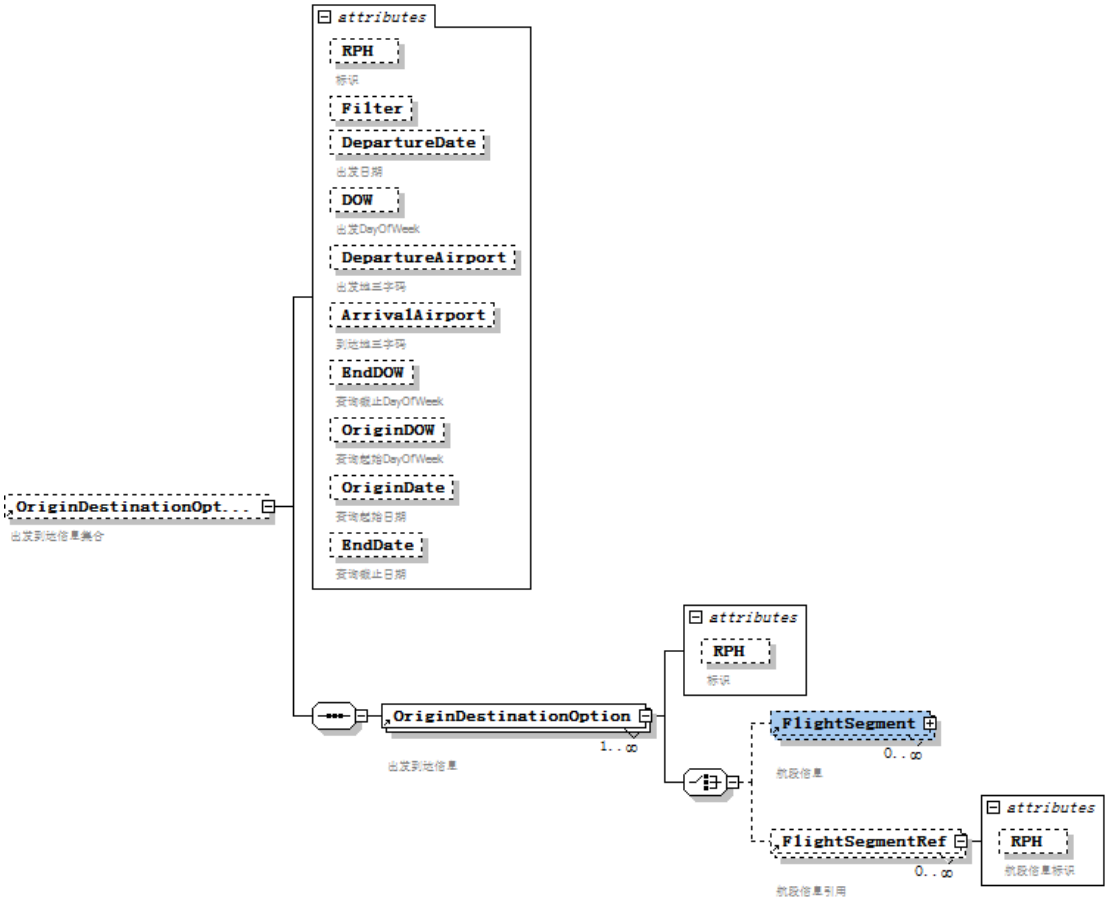
1、POS 节点：



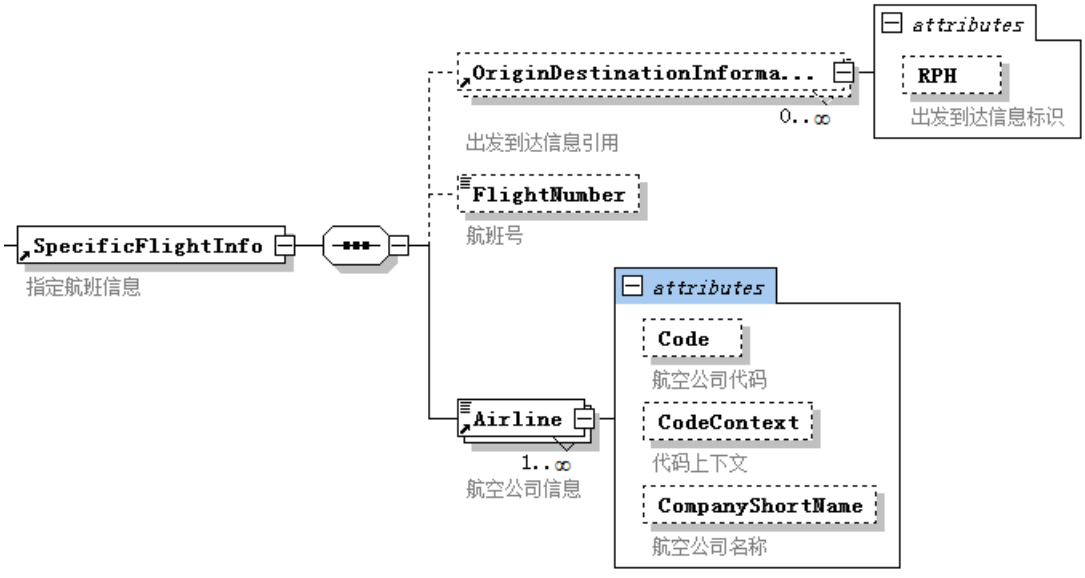
2、OriginDestinationInformation 节点



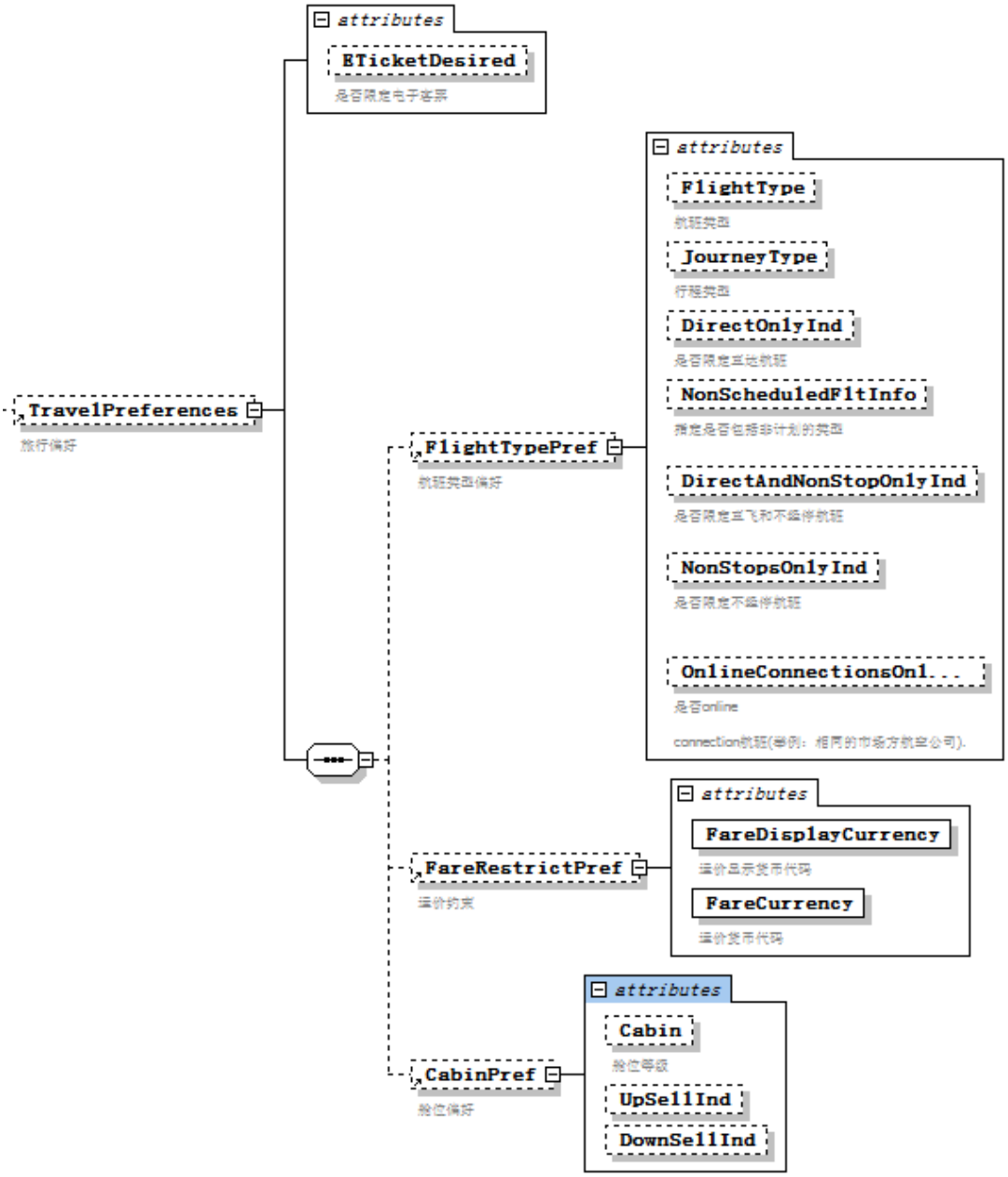
2.1 OriginDestinationOptions 节点



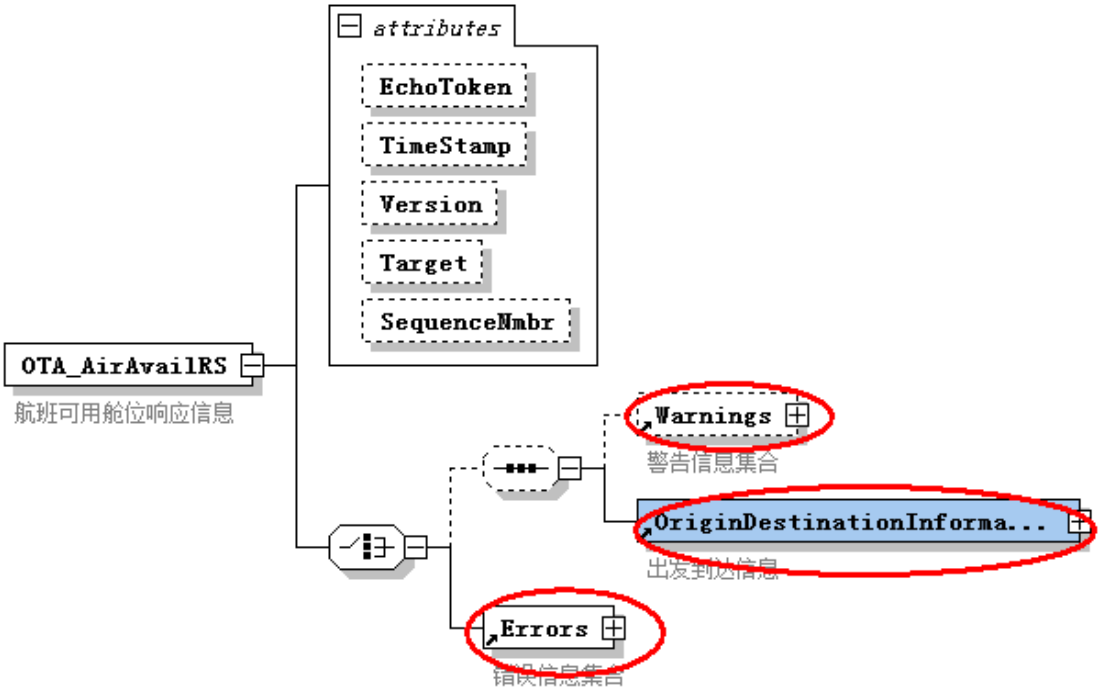
3、SpecificFlightInfo 节点



4、TravelPreferences 节点

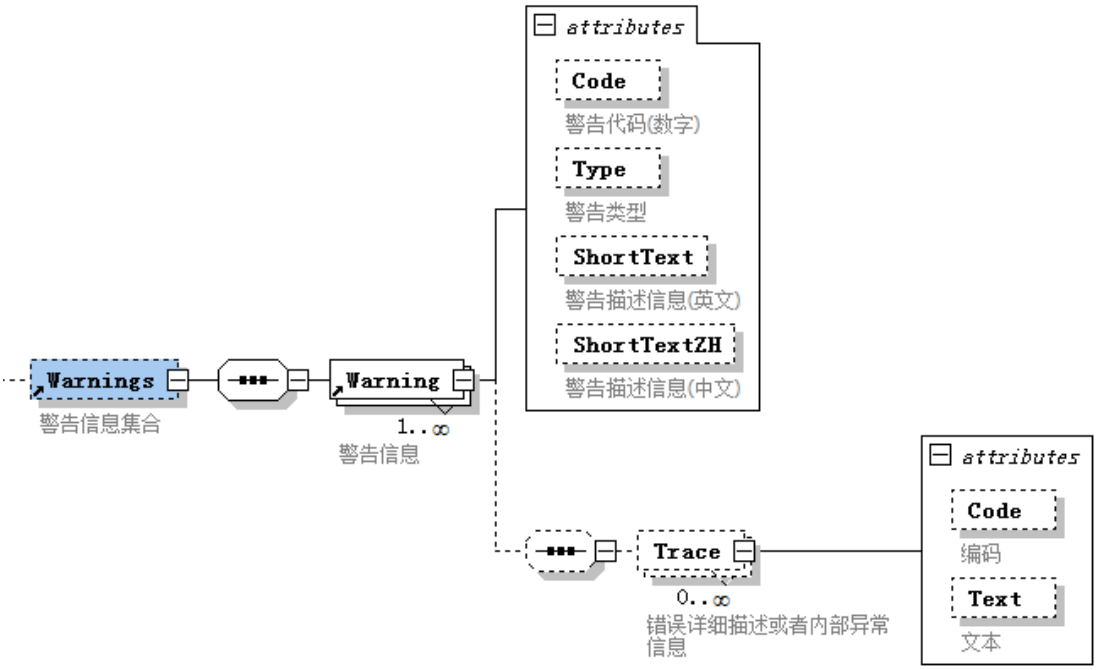

attributes
ETicketDesired
是否限定电子客票
TravelPreferences
旅行偏好
FlightTypePref
航班类型偏好
attributes
FlightType
航班类型
JourneyType
行程类型
DirectOnlyInd
是否限定直达航班
NonScheduledFltInfo
指定是否包括非计划的类型
DirectAndNonStopOnlyInd
是否限定直飞和不经停航班
NonStopsOnlyInd
是否限定不经停航班
OnlineConnectionsOnl...
是否online
connection航班(举例：相同的市场方航空公司).
FareRestrictPref
运价约束
attributes
FareDisplayCurrency
运价显示货币代码
FareCurrency
运价货币代码
CabinPref
舱位偏好
attributes
Cabin
舱位等级
UpSellInd
DownSellInd

5.1.2 响应结构

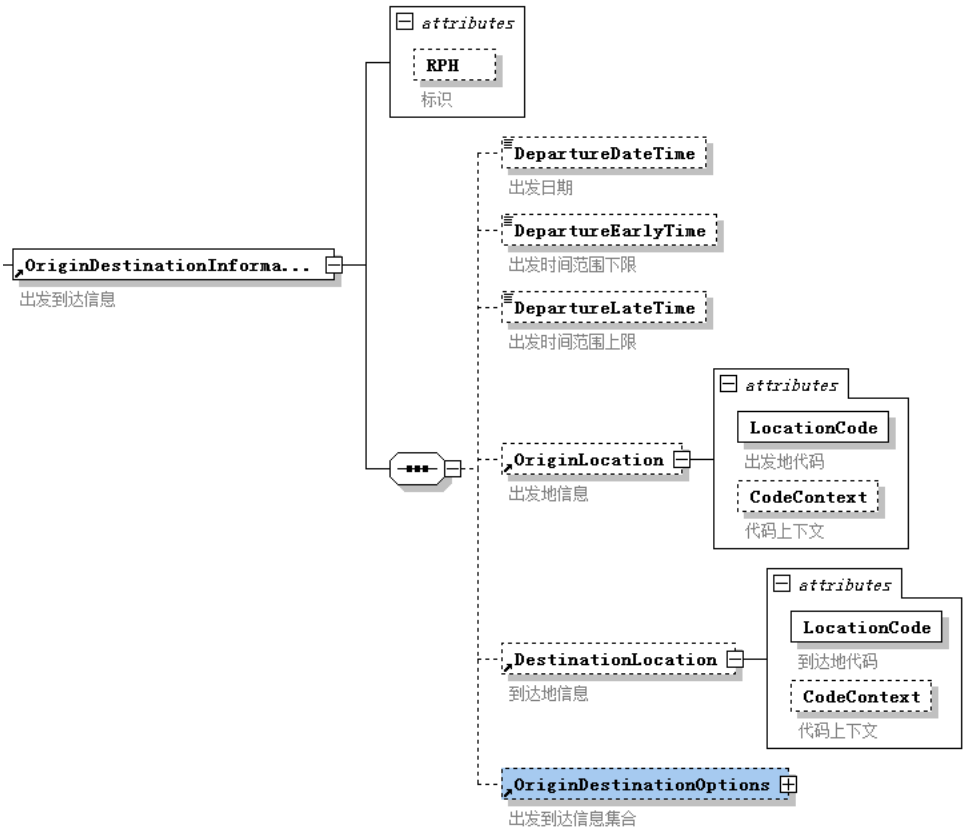


其中标识红色椭圆线的节点结构如下：

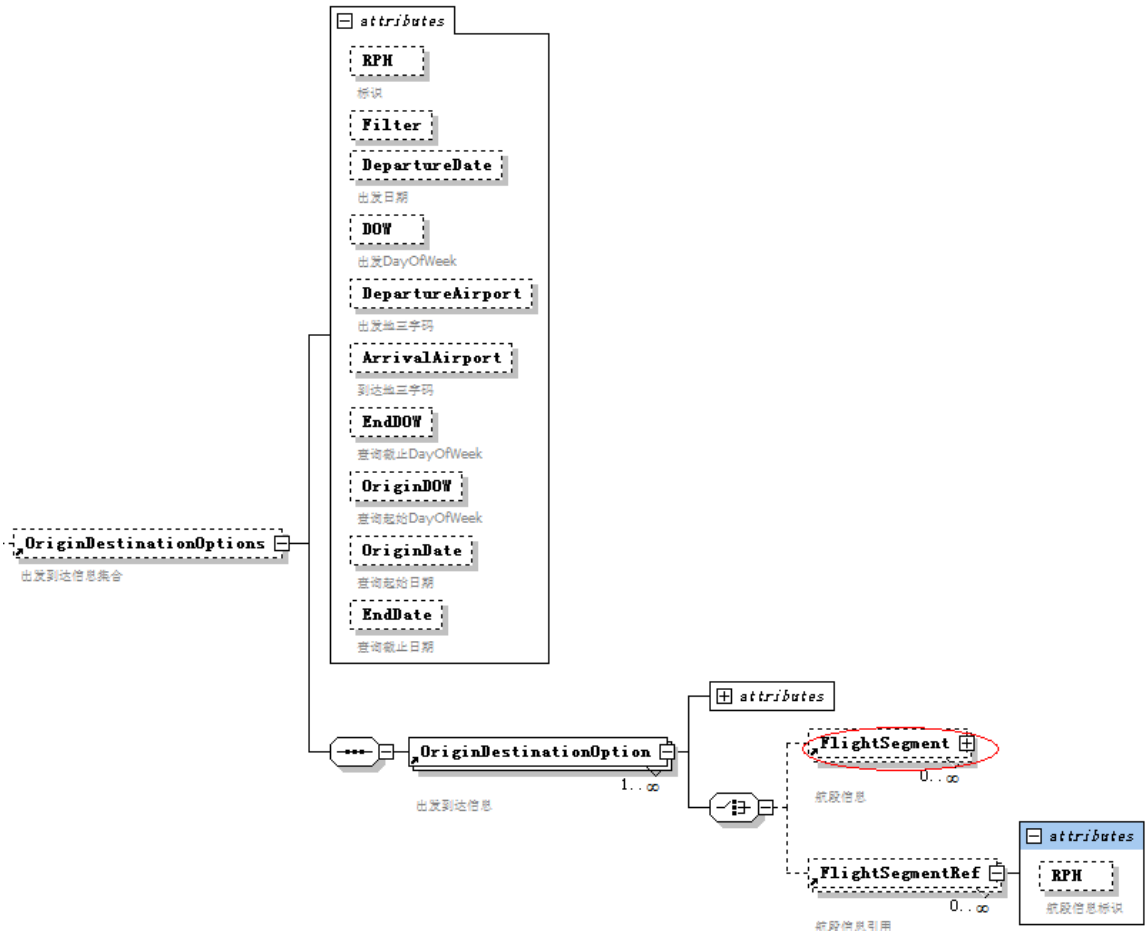
1.Warnings 节点



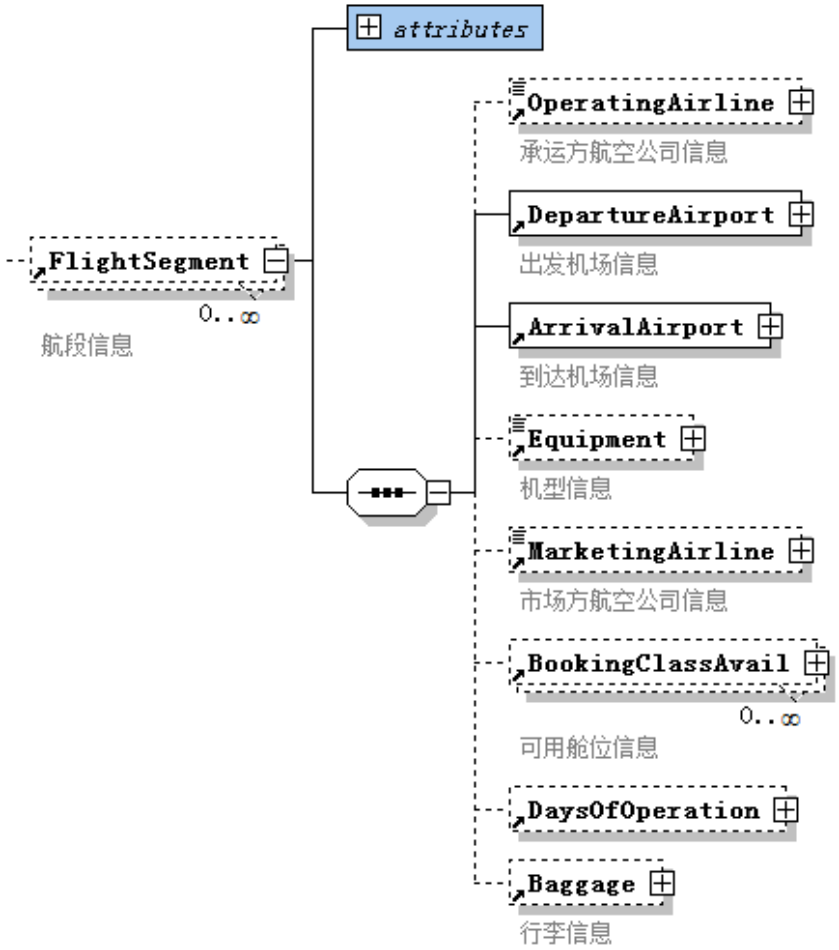
2.OriginDestinationInformation 节点



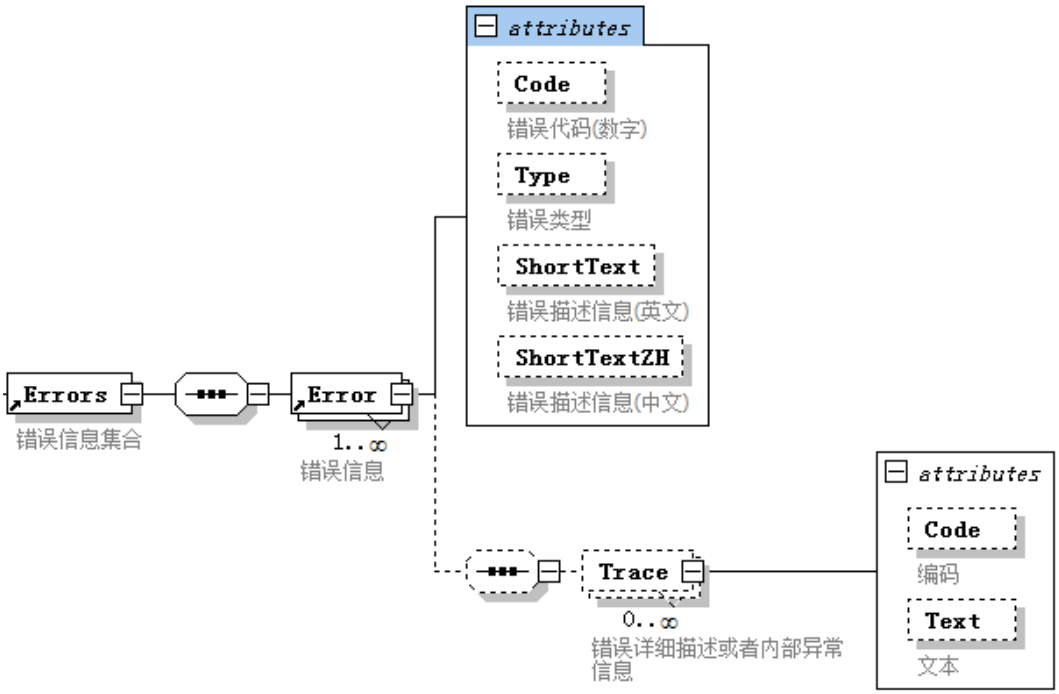
2.1 OrginDestinationOptions 节点



2.1.1 FlightSegment 节点



3.Errors 节点



5.2 请求参数说明

元素名称	说明	Rep	类型	默认值	备注
OTA_AirAvailRQ		1			
POS	零售商信息	1			
Source					
@PseudoCityCode	伪城市代码（Office 代码）	1	String		如：PEK099
OriginDestinationInformation	出发到达信息	1			
@DepartureDateTime	出发日期	1	Date		如：2013-05-22
SpecificFlightInfo	指定航班信息	1			
@FlightNumber	航班号	1	String		如：1831
Airline	航空公司	1			
@Code	航空公司代码	1	String		如：CA

5.3 响应参数说明

元素名称	说明	Rep	类型	默认值	备注
OTA_AirAvailRS		1			
Errors	错误信息	0..1			
Error		0..1			
@Code	错误代码	0..1	String		
@Type	错误类型	0..1	String		
@ShortText	错误详细信息	0..1	String		
Warnings	警告信息	0..1			

Warning		0..1		
@Code	警告代码	0..1	String	
@Type	警告类型	0..1	String	
@ShortText	警告详细信息	0..1	String	
OriginDestinationInformation	出发到达信息			
@DepartureDateTime	出发日期		Date	
OriginLocation	出发地信息			
@LocationCode	出发地三字码		String	
DestinationLocation	到达地信息			
@LocationCode	到达地三字码		String	
OriginDestinationOptions				
OriginDestinationOption				
FlightSegment	航段			
@StopQuantity	经停数		String	
@ASRInd	ASR 标识		String	
@ParticipationLevelCode	连接等级代码		String	
@CodeshareInd	代码共享标识		String	
@FlightNumber	市场方航班号		String	如：1831
DepartureAirport	出发地三字码			
@LocationCode	三字码		String	如：PEK
ArrivalAirport	到达地三字码			
@LocationCode	三字码		String	如：SHA
MarketingAirline	市场方航空公司信息			
@Code	航空公司代码		String	如：CA
BookingClassAvail	可用舱位信息			
@ResBookDesigCode	舱位代码		String	如：K
@ResBookDesigQuantity	可用舱位数量		String	

6.错误说明

6.1http 协议错误

由于接口为 HTTP 接口，HTTP 协议会返回 403 错误：

http 错误代码	说明
403	用户名、密码、IP 错误或无权限

6.2 接口错误

<Errors>节点记录了错误信息。错误信息分为两类：业务错误、系统错误。详细见下表：

系统错误

错误代码	错误描述	说明
-303	inner service error	内部服务异常
-304	sending commands too fast, try again later	访问速度过快,请稍后再试
-305	transaction times limited	访问次数限制

业务错误

错误代码	错误描述	说明
-1	Biz Error	业务异常
-400	format of xml string is not valid	xml 内容格式错误
-402	Input error!	输入参数出错

7.参考样例

7.1 出发时间+航班号+航空公司代码

输入样例：OTA_AirAvailByFltRQ.xml

```
<OTA_AirAvailRQ xmlns="http://espeed.travelsky.com">
	<POS>
		<Source PseudoCityCode="BJS187"/>
	</POS>
	<OriginDestinationInformation>
		<DepartureDateTime>2013-08-19</DepartureDateTime>

	</OriginDestinationInformation>
	<SpecificFlightInfo>
		<FlightNumber>1831</FlightNumber>
		<Airline Code="CA"/>
	</SpecificFlightInfo>

</OTA_AirAvailRQ>
```

输出样例：OTA_AirAvailByFltRSxml

```
<OTA_AirAvailRS>
 <OriginDestinationInformation>
  <DepartureDateTime>2013-08-19</DepartureDateTime>
  <OriginLocation LocationCode="PEK"/>
  <DestinationLocation LocationCode="SHA"/>
```

```
  <OriginDestinationOptions>
    <OriginDestinationOption>
      <FlightSegment StopQuantity="0" ASRInd="false" ParticipationLevelCode="" CodeshareInd="false"
FlightNumber="1831">
        <DepartureAirport LocationCode="PEK"/>
        <ArrivalAirport LocationCode="SHA"/>
        <MarketingAirline Code="CA"/>
        <BookingClassAvail ResBookDesigCode="F" ResBookDesigQuantity="L"/>
        <BookingClassAvail ResBookDesigCode="A" ResBookDesigQuantity="L"/>
        <BookingClassAvail ResBookDesigCode="O" ResBookDesigQuantity="L"/>
        <BookingClassAvail ResBookDesigCode="W" ResBookDesigQuantity="A"/>
        <BookingClassAvail ResBookDesigCode="Y" ResBookDesigQuantity="A"/>
        <BookingClassAvail ResBookDesigCode="B" ResBookDesigQuantity="A"/>
        <BookingClassAvail ResBookDesigCode="M" ResBookDesigQuantity="A"/>
        <BookingClassAvail ResBookDesigCode="H" ResBookDesigQuantity="A"/>
        <BookingClassAvail ResBookDesigCode="K" ResBookDesigQuantity="A"/>
        <BookingClassAvail ResBookDesigCode="L" ResBookDesigQuantity="A"/>
        <BookingClassAvail ResBookDesigCode="Q" ResBookDesigQuantity="S"/>
        <BookingClassAvail ResBookDesigCode="G" ResBookDesigQuantity="S"/>
        <BookingClassAvail ResBookDesigCode="S" ResBookDesigQuantity="A"/>
        <BookingClassAvail ResBookDesigCode="X" ResBookDesigQuantity="5"/>
        <BookingClassAvail ResBookDesigCode="N" ResBookDesigQuantity="3"/>
        <BookingClassAvail ResBookDesigCode="V" ResBookDesigQuantity="S"/>
        <BookingClassAvail ResBookDesigCode="U" ResBookDesigQuantity="A"/>
        <BookingClassAvail ResBookDesigCode="T" ResBookDesigQuantity="S"/>
        <BookingClassAvail ResBookDesigCode="E" ResBookDesigQuantity="S"/>
      </FlightSegment>
    </OriginDestinationOption>
  </OriginDestinationOptions>
```

```
 </OriginDestinationInformation>
</OTA_AirAvailRS>
```

8.接口调用

航班可用舱位查询服务提供基于 XML 文档规范的标准 HTTP 协议接口，通过在标准 HTTP 通道上传输 XML 格式的请求和数据来完成服务调用。服务的输入输出定义在标准的 XSD 文档中，用户根据自有系统的实现环境结合 XSD 文档进行数据到对象间的转换。在线分销平台提供服务接口 URL,用户使用 HTTP 协议按照服务定义正确构造请求输入，正确解析数据输出实现服务的调用。

8.1 接口申请

请与您所在城市的中国航信分支机构联系，资质审查通过后签署试用协议，即可获得试用接口。

8.2 接口认证

授权用户调用试用接口时需要使用已分配的账号和密码、并使用申请时的 IP。如果已变化 IP 请联系业务人员申请变更 IP。

8.3 接口地址

接口描述位置：http://espeed.travelsky.com/develop/xml/AirAvailByFlt?xsd

接口试用地址：http://espeed.travelsky.com/develop/xml/AirAvailByFlt

8.4 JAVA 调用示例

Java 调用 http 接口示例如下：

其中该样例程序只是为了展示如何接入使用

```
import java.io.BufferedReader;
```

```
import java.io.ByteArrayOutputStream;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.zip.GZIPInputStream;

import java.util.zip.GZIPOutputStream;


import org.apache.commons.httpclient.Credentials;

import org.apache.commons.httpclient.DefaultHttpMethodRetryHandler;

import org.apache.commons.httpclient.HttpClient;

import org.apache.commons.httpclient.HttpState;

import org.apache.commons.httpclient.HttpStatus;

import org.apache.commons.httpclient.UsernamePasswordCredentials;

import org.apache.commons.httpclient.auth.AuthScope;

import org.apache.commons.httpclient.methods.ByteArrayRequestEntity;

import org.apache.commons.httpclient.methods.PostMethod;

import org.apache.commons.httpclient.methods.RequestEntity;

import org.apache.commons.httpclient.params.HttpMethodParams;


/**

 * http 客户端调用示例(数据压缩版)

 */

public class HttpClientDemo {


	public static void main(String[] args) {


		//实际使用时，请设置正确的变量值

		//用户名

		String username = "用户名";

		//密码
```

```
String pwd = "密码";
//服务地址
String serviceUrl = "服务地址";
//请求 XML
String reqStr = "test-xml";

// 构造 HttpClient 的实例
HttpClient httpClient = new HttpClient();
//调用验证信息
HttpState state = new HttpState();
Credentials credentials = new UsernamePasswordCredentials(username, pwd);
state.setCredentials(AuthScope.ANY, credentials);
httpClient.setState(state);

// 创建 POST 方法的实例
PostMethod postMethod = new PostMethod(serviceUrl);
// 使用系统提供的默认的恢复策略
postMethod.getParams().setParameter(HttpMethodParams.RETRY_HANDLER,
        new DefaultHttpMethodRetryHandler());
try {
    // 请求参数的数据压缩
    ByteArrayOutputStream out = new ByteArrayOutputStream();
    GZIPOutputStream gzip = new GZIPOutputStream(out);
    if (reqStr != null && !"".equals(reqStr)) {
        gzip.write(reqStr.getBytes());
    }
    gzip.close();
    RequestEntity requestEntity = new ByteArrayRequestEntity(out
            .toByteArray());
```

```
postMethod.setRequestEntity(requestEntity);
postMethod.addRequestHeader("Content-Type",
                "text/html;charset=UTF-8");
postMethod.addRequestHeader("accept-encoding", "gzip");
postMethod.addRequestHeader("content-encoding", "gzip");

// 执行 getMethod
int statusCode = httpClient.executeMethod(postMethod);
if (statusCode != HttpStatus.SC_OK) {
        throw new Exception("Invoke Get Method Failed, HttpStatus = "
                        + statusCode);
}

// 返回结果的数据解压
InputStream is = postMethod.getResponseBodyAsStream();
BufferedReader br = new BufferedReader(new InputStreamReader(
                new GZIPInputStream(is)));

StringBuffer respStr = new StringBuffer();
String line = null;
while ((line = br.readLine()) != null) {
        respStr.append(line);
}

// 打印结果
System.out.println(respStr.toString());

out.flush();
out.close();
```

```
                is.close();
            } catch (Exception e) {
                // 发生致命的异常，可能是协议不对或者返回的内容有问题
                e.printStackTrace();
            } finally {
                // 释放连接
                postMethod.releaseConnection();
            }

    }
}
```

8.5.NET 调用示例

.Net 调用 http 接口示例如下：

其中该样例程序只是为了展示如何接入使用

(1)建立一个 cs 文件,与服务器通信的类：

```
// 定义 xml 与服务器的通信
class OTA_XMLHTTP
 {
        string user;        //用户名
        string pwd;     //密码
         string url;   //服务地址
        public OTA_XMLHTTP(string _user,string _pwd,string _url)
        {
            user = _user;
            pwd = _pwd;
            url = _url;
        }
```

```
// 获取服务器返回的字符串
public string GetResponse(string requestXml)
{
    string xmlString = "";     //要返回的 xml 字符串
    bool a= requestXml.Contains("<?xml version="1.0" encoding="UTF-8"?>");
    requestXml = requestXml.Replace("<?xml version="1.0" encoding="UTF-8"?>\n", "");
    try
    {
        //制定服务器地址
        HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(url);
        request.Method = WebRequestMethods.Http.Post; //设定 http 的传递方式
        request.ContentType = "application/x-www-form-urlencoded";
       //设定 http 的 header
       if (user != null && pwd != null)
       {
            string user_pwd = user + ":" + pwd;
            byte[] authBytes = Encoding.UTF8.GetBytes(user_pwd.ToCharArray());
            request.Headers.Add("Authorization", "Basic " + Convert.ToBase64String(authBytes));
           request.Headers.Add("Content-Encoding", "gzip");
           request.Headers.Add("Accept-Encoding", "gzip");
        }
       //将数据写入流中
        string para = "request";
        ASCIIEncoding encoding = new ASCIIEncoding();
        byte[] data = encoding.GetBytes(para + " = " + requestXml);
        request.ContentLength = data.Length;
        Stream stream = request.GetRequestStream();
        stream.Write(data, 0, data.Length);
        stream.Close();
```

```
                //建立获取 http 返回的 response
                HttpWebResponse response = (HttpWebResponse)request.GetResponse();
                //读取 http 返回的字符串
            Stream responseStream = response.GetResponseStream();
            if (response.ContentEncoding.ToLower().Contains("gzip"))
                  responseStream = new GZipStream(responseStream, CompressionMode.Decompress);
            else if (response.ContentEncoding.ToLower().Contains("deflate"))
                  responseStream = new DeflateStream(responseStream, CompressionMode.Decompress);
            StreamReader streamReader = new StreamReader(responseStream, Encoding.UTF8);
                   xmlString = streamReader.ReadToEnd();
                 streamReader.Close();
           }
           catch (WebException e)
           {
               xmlString = e.Message;
           }
         return xmlString;
      }
   }
```

（2）调用上面的文件

调用方法：

```
          string user="用户名";
        string pwd="密码";
        string url="接口地址″;
        OTA_XMLHTTP xmlHttp = new OTA_XMLHTTP(user, pwd, url);
        PonseXmlBox.Text = xmlHttp.GetResponse(requestXml);
```